Compte rendu des travaux pratiques du cours de Java avancé Premier TP N°1

Réalisé par : ***Intissar Lobad***

Encadré par : ***Madame Asmaa Elkourchi***

2024-2025

Table des matières

1. Architecture du Projet (MVC + DAO)

- **MVC** :
  - **Model** : Représente la classe métier `Employee` avec ses attributs, constructeurs et énumérations (`Role`, `Poste`).
  - **View** : Représente les interfaces utilisateur qui communiquent avec le contrôleur.
  - **Controller** : Interagit entre la `View` et le `Model` en appelant les méthodes de DAO.
- **DAO** (Data Access Object) : Gère les interactions avec la base de données via `EmployeeDAOImpl`.

2. Fonctionnalités Analytiques

a) Ajouter un Employé (ajouterEmployee)

Étapes :

1. **Récupération des données** : Les données de l'employé sont passées via un objet `Employee`.
2. **Connexion à la base** : La méthode `DBConnection.getConnection()` est appelée pour établir la connexion.
3. **Requête SQL** :
   - La requête `INSERT INTO` est utilisée pour ajouter les informations de l'employé avec des placeholders (`?`).
   - `roleId` et `posteId` sont récupérés via des sous-requêtes SQL utilisant les noms des rôles et des postes.
4. **Exécution** :
   - La requête est exécutée via `executeUpdate()` pour effectuer l'insertion.
   - Une vérification est faite pour confirmer si l'ajout a réussi.

Code :

```
public void ajouterEmployee(Employee employee) {
    String query = "INSERT INTO employe (nom, prenom, email, telephone,
salaire, roleId, posteId) " +
                   "VALUES (?, ?, ?, ?, ?, (SELECT id FROM role WHERE
nom=?), (SELECT id FROM poste WHERE nom=?))";
    try (PreparedStatement stmt =
DBConnection.getConnection().prepareStatement(query)) {
        stmt.setString(1, employee.getNom());
        stmt.setString(2, employee.getPrenom());
        stmt.setString(3, employee.getEmail());
        stmt.setString(4, employee.getTelephone());
        stmt.setDouble(5, employee.getSalaire());
        stmt.setString(6, employee.getRole().name());
        stmt.setString(7, employee.getPoste().name());
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }}
```

b) Afficher les Employés (afficherEmployees)

Étapes :

1. **Connexion** : Appel de `DBConnection.getConnection()` pour accéder à la base.
2. **Requête SQL** :
   - Une jointure `JOIN` entre les tables `employe`, `role` et `poste` permet de récupérer toutes les informations nécessaires.
3. **Exécution** :
   - La requête est exécutée via `executeQuery()` pour obtenir un `ResultSet`.
4. **Mapping des Résultats** :
   - Les données de chaque ligne sont converties en objets `Employee` et ajoutées à une `List<Employee>`.
5. **Retour** : La liste des employés est retournée.

Code :

```
public List<Employee> afficherEmployees() {
    List<Employee> employees = new ArrayList<>();
    String query = "SELECT e.id, e.nom, e.prenom, e.email, e.telephone,
e.salaire, r.nom AS roleNom, p.nom AS posteNom " +
                   "FROM employe e JOIN role r ON e.roleId = r.id JOIN
poste p ON e.posteId = p.id";
    try (PreparedStatement stmt =
DBConnection.getConnection().prepareStatement(query)) {
        ResultSet rs = stmt.executeQuery();
        while (rs.next()) {
            Employee emp = new Employee(rs.getInt("id"),
rs.getString("nom"), rs.getString("prenom"),
                        rs.getString("email"), rs.getString("telephone"),
rs.getDouble("salaire"),
                        Role.valueOf(rs.getString("roleNom")),
Poste.valueOf(rs.getString("posteNom")), 0);
            employees.add(emp);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return employees;
}
```

c) Modifier un Employé (modifierEmployee)

Étapes :

1. **Connexion** : Accès à la base via `DBConnection.getConnection()`.
2. **Requête SQL** :
   - La requête `UPDATE` met à jour les informations de l'employé sélectionné par son `id`.
   - Les nouveaux `roleId` et `posteId` sont récupérés avec des sous-requêtes SQL.
3. **Paramétrage** :
   - Les nouvelles valeurs sont assignées via `stmt.setXXX()`.
4. **Exécution** : `executeUpdate()` met à jour l'enregistrement.
5. **Confirmation** : Une vérification est effectuée pour s'assurer de la réussite de la mise à jour.

Code :

```
public void modifierEmployee(int id, Employee modifiedEmployee) {
    String query = "UPDATE employe SET nom=?, prenom=?, email=?,
telephone=?, salaire=?, " +
                   "roleId=(SELECT id FROM role WHERE nom=?),
posteId=(SELECT id FROM poste WHERE nom=?) WHERE id=?";
    try (PreparedStatement stmt =
DBConnection.getConnection().prepareStatement(query)) {
        stmt.setString(1, modifiedEmployee.getNom());
        stmt.setString(2, modifiedEmployee.getPrenom());
        stmt.setString(3, modifiedEmployee.getEmail());
        stmt.setString(4, modifiedEmployee.getTelephone());
        stmt.setDouble(5, modifiedEmployee.getSalaire());
        stmt.setString(6, modifiedEmployee.getRole().name());
        stmt.setString(7, modifiedEmployee.getPoste().name());
        stmt.setInt(8, id);
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

d) Supprimer un Employé (supprimerEmployee)

Étapes :

1. **Connexion** : Appel de `DBConnection.getConnection()`.
2. **Requête SQL** :
    - La requête `DELETE` supprime un employé à partir de son `id`.
3. **Exécution** : La suppression est réalisée via `executeUpdate()`.
4. **Confirmation** : Une vérification garantit la suppression effective.

Code :

```
public void supprimerEmployee(int id) {
    String query = "DELETE FROM employe WHERE id=?";
    try (PreparedStatement stmt =
DBConnection.getConnection().prepareStatement(query)) {
        stmt.setInt(1, id);
        stmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

3. Conclusion

Le projet suit une architecture **MVC + DAO** structurée :

- **Model** : Classe `Employee`.
- **DAO** : `EmployeeDAOImpl` pour la gestion des opérations CRUD.
- **View/Controller** : Les vues interagissent avec le contrôleur qui appelle les méthodes DAO.

Pour accéder au code source complet du projet, vous pouvez consulter ce lien : Intysar/gestionDesEmployes